guida rapida al C64

LEGENDA

Elenchiamo le convenzioni usate nel definire la sintassi di istruzioni, comandi e funzioni BASIC:

[ ]	Racchiude gli elementi non indispensabili
v	Nome di variabile numerica, ad esempio X, X%, VETTORE (3, 5)
v$	Nome di variabile-stringa, ad esempio P$, NOME$ (4, 2)
i, j	Numeri o variabili intere, ad esempio 93, J%
n	Numero decimale, sotto forma di numero, variabile o espressione numerica, ad esempio 42.867, NUM, M (3, 7), SIN (N*COS(B))
c	Espressione logica (relazione), ad esempio B>=A, A$=B$
a$	Stringa, variabile-stringa o espressione-stringa, ad esempio M$, "PIPPO", A$+RIGHT$(M$, 3)
ln	Numero di linea in un programma
ind	Locazione di memoria, indicata da un numero compreso tra 0 e 65535
d	Numero del dispositivo di input/output
prg	Nome di programma
msg	Messaggio da stampare; può essere costituito da stringhe, numeri, variabili, espressioni, ecc.
...	Gli elementi che si trovano prima dei puntini possono essere ripetuti quante volte sia necessario

---

ISTRUZIONI BASIC

Controllo del programma

END	Arresta l'esecuzione del programma
FOR v=n1 TO n2 [STEP n3]	Apre un ciclo indicizzato con v, costituito dalle istruzioni che si trovano prima di NEXT v. Ogni ciclo aggiunge n3 (1 se STEP viene omesso) a n1, continuando finché n1>n2, oppure n1<n2 nel caso n3 sia negativo
GOSUB ln	Esegue la subroutine che inizia in ln, proseguendo finché non viene incontrato un RETURN
GOTO ln	L'esecuzione del programma prosegue in ln
IF c [THEN] [GOTO] ln	Se la condizione è vera continua in ln, altrimenti passa alla linea successiva
IF c THEN istruzioni	Se la condizione è vera esegue le istruzioni, altrimenti passa alla linea successiva
NEXT [v[,v...]]	Chiude uno o più cicli FOR...NEXT
ON v GOTO ln 1 [,ln 2...]	Se v=1 il controllo passa a ln1, se v=2 passa a ln2 e così via; se v=0 oppure v è maggiore del numero di ln elencati, passa alla linea successiva
ON v GOSUB ln 1 [, ln 2...]	Come sopra, ma il controllo passa a delle subroutine
RETURN	Conclude una subroutine. Il controllo passa all'istruzione successiva all'ultimo GOSUB
STOP	Arresta l'esecuzione del programma visualizzando un messaggio di BREAK
SYS ind	Passa il controllo a una routine in linguaggio macchina che inizia in ind
WAIT ind, i [,j]	Esegue un AND tra il contenuto di ind e i; se j è specificato, sul valore ottenuto viene eseguito un OR esclusivo (EOR) con j. Finché il risultato è zero, l'esecuzione del programma resta bloccata

Manipolazione dei dati e assegnamenti

CLR	Cancella variabili e vettori di qualsiasi tipo
DATA n,a$...	Immagazzina dati numerici o stringhe che debbano essere utilizzate durante l'esecuzione del programma; le stringhe devono essere racchiuse tra apici (") se contengono virgole, due punti, spazi o caratteri shiftati
DEF FN v1 (v2) = n	Definisce la funzione v1 di argomento v2; l'espressione n può contenere v2

DIM v1 (i[,j...]) [,v2...]	Definisce uno o più vettori a una o più dimensioni, contenenti (i+1)*(j+1)*... elementi
[LET] v=	assegna alla variabile v il valore che segue =
POKE ind,i	Assegna il valore i (0–255) alla locazione ind
READ v1[,v2...]	Assegna alla variabile v1 il valore contenuto nella istruzione DATA puntata in quel momento, assegna a v2 il valore successivo, e così via
RESTORE	Azzera il puntatore delle DATA, in modo che indichi la prima istruzione DATA contenuta nel programma
TI [ME]$=a$	Assegna al timer interno del C64 il valore di a$, che deve essere una stringa di 6 cifre nel formato HHMMSS

Input/Output

CLOSE i	Chiude il file logico numero i (1–255) precedentemente aperto con una istruzione OPEN, inviando al dispositivo di I/O il segnale di fine file
CMD i[,a$]	Invia al dispositivo di output, sul quale è stato aperto il file logico i, tutti i messaggi che normalmente verrebbero diretti allo schermo. Eventualmente scrive nel file (e quindi invia al dispositivo) la stringa a$
GET v1[,v2...]	Legge e assegna a v1 il tasto attualmente premuto (qualsiasi se v1 è una variabile-stringa, solo i tasti numerici se v1 è numerica). Eventualmente prosegue con v2, ecc.
GET#i,v1[,v2...]	Come sopra, ma preleva il carattere dal file numero i
INPUT ["msg";]v1[,v2...]	Stampa un punto interrogativo (?) dopo l'eventuale messaggio, visualizza il cursore ed attende che da tastiera venga impostato il valore da assegnare a v1, finché non venga premuto il tasto RETURN. Se sono specificate anche altre variabili (v2, ecc.) i valori devono venir digitati separati da virgole
INPUT#i,v1[,v2...]	Preleva uno o più valori dal file logico i, e li assegna alle variabili v1, v2, ecc. I valori devono essere separati nel file da CHR$ (13), virgola, punto e virgola o due punti
OPEN i[,d,j,a$]	Apre il file logico i, sul dispositivo d (default 1 = registratore) e con indirizzo secondario j (default 0 = lettura), chiamandolo a$
PRINT msg [;] [,] [msg]...	Stampa sullo schermo i messaggi elencati, che possono essere separati da virgola o punto e virgola
PRINT#i,msg[;] [,] [msg]...	Come sopra, ma l'output è diretto al file i, e quindi al dispositivo sul quale esso è stato aperto

Documentazione

REM	Note inserite nel programma, che vengono ignorate durante l'esecuzione

COMANDI BASIC

CONT	Continua l'esecuzione del programma dopo che essa era stata arrestata da istruzioni STOP, END o tramite il tasto RUN/STOP
LIST	Lista l'intero programma in memoria
LIST ln	Lista la linea ln
LIST ln –	Lista tutte le linee da ln alla fine del programma
LIST –ln	Lista tutte le linee dall'inizio del programma a ln
LIST ln1–ln2	Lista le linee comprese tra ln1 e ln2
LOAD	Carica il primo programma che viene trovato sul nastro (se il comando è inserito in una linea di programma esegue anche il RUN)
LOAD "prg"[,d]	Cerca il programma chiamato prg su nastro (default d=1) oppure su disco (d=8); se lo trova effettua il caricamento
LOAD "prg",d,1	Come sopra, ma il programma viene caricato nell'area di memoria che occupava quando era stato salvato (comando usato per linguaggio macchina e dati)

NEW	Cancella il programma attualmente in memoria e le sue variabili
RUN [ln]	Esegue il programma in memoria (eventualmente a partire dalla linea ln)
SAVE	Salva su nastro il programma in memoria senza assegnargli un nome
SAVE "prg"[,d]	Salva il programma in memoria su nastro (default d=1) oppure su disco (d=8) chiamandola prg
SAVE "prg",d,i,	Come sopra, ma viene specificato l'indirizzo secondario i. Se i=1 viene memorizzata anche la locazione a partire dalla quale verrà effettuato il caricamento, se i=2 viene aggiunto un segnale di fine nastro (solo con d=1), e infine se i=3 vengono eseguite entrambe le opzioni precedenti
VERIFY	Confronta il primo programma trovato su nastro con quello in memoria
VERIFY"prg",d	Cerca il programma prg sul dispositivo d e lo confronta con quello in memoria

FUNZIONI BASIC

ABS(n)	Valore assoluto di n; ad esempio, ABS(5)=5 e ABS(−8)=8
ASC(a$)	Codice CBM ASCII del primo carattere di a$
ATN(n)	Angolo in radianti la cui tangente è n
CHR$(i)	Carattere avente il codice CBM ASCII i (0−255)
COS(n)	Coseno dell'angolo n (n in radianti)
EXP(n)	Numero di Nepero (e=2.71827...) elevato alla potenza n-esima (antilogaritmo di n)
FNv1(n)	Risultato della funzione FNv1 precedentemente definita, avente come argomento n
FRE(0)	Numero di byte liberi per i programmi BASIC (aggiungere 65535 se il risultato è negativo)
INT(n)	Arrotondamento di n all'unità per difetto; ad esempio INT (3.7)=3, mentre INT(−3.7)=−4
LEFT$(a$,i)	Stringa composta dagli ultimi i caratteri a sinistra di a$
LEN(a$)	Numero di caratteri che formano a$
LOG(n)	Logaritmo naturale (in base e) di n
MID$(a$,i,j)	Stringa composta da j caratteri presi a partire dall'i-esimo carattere di a$. Se j viene omesso la stringa risultante conterrà tutti i caratteri dall'i-esimo sino alla fine di a$
PEEK(ind)	Valore (0−255) contenuto nella locazione ind
POS(0)	Colonna in cui si trova attualmente il cursore (0-79 in quanto vengono considerate due righe dello schermo per volta)
RIGHT$(a$,i)	Stringa composta dagli ultimi i caratteri a destra di a$
RND(n)	Numero pseudo-casuale prelevato da una sequenza e compreso tra 0 e 1. La sequenza viene costruita a partire da un seme, che viene ricavato dal clock interno se n=0, mentre è lo stesso n se n<0. Se invece n>0 il numero pseudo-casuale viene prelevato dalla sequenza attuale
SGN(n)	Segno di n. Vale 0 se n=0, 1 se n>0 e −1 se n<0
SIN(n)	Seno dell'angolo n (n in radianti)
SPC(i)	In una istruzione PRINT fa avanzare il cursore di i posizioni
SQR(n)	Radice quadrata di n (n>=0)
ST[ATUS]	Valore del byte di stato dopo operazioni di I/O
STR$(n)	Stringa formata dalle cifre che compongono n, precedute da uno spazio se n>0 o dal segno meno se n<0
TAB(i)	In una istruzione PRINT sposta il cursore alla colonna i+1 (0−79) della riga dove esso si trova attualmente. Se il cursore ha superato tale colonna, stampa di seguito
TAN(n)	Tangente dell'angolo n (n in radianti)
TI[ME]	Valore del clock interno: tempo in sessantesimi di secondo trascorso dall'accensione del computer o dall'ultimo assegnamento di TI$
TI[ME]$	Stringa di 6 caratteri che costituisce un orologio con ciclo di 24 ore, nel formato HHMMSS, ricavato dal clock interno
USR(n)	Chiama una routine in linguaggio macchina, passandole n come argomento. L'indirizzo iniziale della routine deve venir memorizzato nelle locazioni 785-786
VAL(a$)	Valore numerico della stringa a$. Risulta diverso da zero solo se il

primo carattere è costituito da una cifra, dal punto oppure dai segni + e − (ad esempio, VAL("8.3")=8.3, VAL("ABC")=0, VAL("−31ZZ")=−31)

VARIABILI BASIC

I nomi delle variabili sono costituiti da una lettera seguita da un qualsiasi numero di caratteri; tuttavia solo i primi due caratteri del nome vengono riconosciuti dall'Interprete BASIC. Il tipo della variabile viene identificato tramite un eventuale suffisso (l'ultimo carattere):

Tipo numerico (decimale), senza suffisso (ad esempio AB=3.86). Può variare in valore assoluto tra 2.9E−39 e 1.7E+38
Tipo numerico (intero), con suffisso % (ad esempio AB%=527). Può variare tra −32768 e 32767
Tipo stringa, con suffisso $ (ad esempio AB$="PIPPO"). Può essere composto da 0 a 255 caratteri

I vettori (insiemi di variabili) possono avere qualsiasi numero di dimensioni ed elementi. Se un vettore viene usato senza averne indicato le dimensioni tramite una istruzione DIM, ad esso vengono assegnati 11 elementi per ogni dimensione.
Ad esempio, se poniamo nel nostro programma l'istruzione A(2,0)=57, il vettore A(i,j) viene automaticamente dimensionato con DIM A(10,10)

CARATTERI SPECIALI

" Delimita le stringhe
$ Identifica le variabili-stringa
% Identifica le variabili intere
In un'istruzione PRINT, fa avanzare il cursore sino all'inizio della prossima zona; ogni riga di 80 caratteri è divisa in 8 zone larghe 10 caratteri (tabulatura automatica)
: Delimita le istruzioni all'interno di una linea di programma
; In un'istruzione PRINT, fa restare il cursore nella posizione raggiunta dopo la stampa dell'ultimo messaggio
= Assegna un valore ad una variabile (ad esempio [LET]K=4)
? Abbreviazione di PRINT
E Indica l'esponente in notazione scientifica (significa "per 10 elevato a"). Ad esempio, 1.7E-6 equivale a 0.0000017

OPERATORI

Operatori aritmetici

simbolo	operazione	priorità
+	addizione	4
−	sottrazione	4
*	moltiplicazione	3
/	divisione	3
−	meno (ad es. −8)	2
↑	elev. a potenza	1

Operatori relazionali

simbolo	operazione	priorità
=	uguale a	5
<	minore di	5
>	maggiore di	5
<=	minore o uguale a	5
>=	magg. o uguale a	5
<>	diverso da	5

Operatori logici

simbolo	operazione	priorità
OR	somma logica	8
AND	prodotto logico	7
NOT	negazione logica	6

NOTA: Gli operatori logici possono agire su relazioni (come A>0) o anche su variabili numeriche. Se la relazione è vera essa vale −1, mentre se è falsa vale 0. Per quanto riguarda le variabili numeriche, invece, lo zero viene interpretato come falso mentre qualsiasi valore non nullo viene interpretato come vero. Quindi IF A<>0 THEN... equivale a IF A THEN...

Tavole della verità

vero AND vero =vero
vero AND falso =falso
falso AND vero =falso
falso AND falso =falso

vero OR vero =vero
vero OR falso =vero
falso OR vero =vero
falso OR falso =falso

vero EOR vero =falso
vero EOR falso =vero
falso EOR vero =vero
falso EOR falso =falso

NOT vero =falso
NOT falso =vero

MESSAGGI DI ERRORE

BAD DATA	Un'istruzione DATA conteneva una stringa mentre era richiesto un numero
BAD SUBSCRIPT	L'elemento di vettore specificato non esiste (fuori dal dimensionamento)
BREAK IN ln	Il tasto RUN/STOP è stato premuto durante l'esecuzione della linea ln
CAN'T CONTINUE	L'esecuzione del programma non può essere ripresa usando il comando CONT
DEVICE NOT PRESENT	Il dispositivo di I/O che si è cercato di usare non è collegato
DIVISION BY ZERO	Si è richiesta un'operazione aritmetica impossibile
EXTRA IGNORED	Durante l'esecuzione di una INPUT si sono introdotti troppi dati, oppure un dato conteneva una virgola o due punti
FILE NOT FOUND	Il registratore è giunto alla fine del nastro senza trovare il file richiesto, oppure tale file non è presente sul disco inserito nel drive
FILE NOT OPEN	Si è cercato di compiere operazioni di I/O usando un file non aperto
FILE OPEN	Si è cercato di aprire con una OPEN un file già aperto
FORMULA TOO COMPLEX	Troppi livelli di parentesi di un'espressione
ILLEGAL DIRECT	Si è cercato di eseguire in modo diretto istruzioni che sono utilizzabili solo all'interno di un programma (INPUT, DATA, ecc.)
ILLEGAL QUANTITY	L'argomento numerico di una funzione è fuori dall'intervallo consentito
LOAD	Problemi nel caricamento di un programma da nastro
NEXT WITHOUT FOR	Durante l'esecuzione del programma viene incontrato un NEXT non preceduto dal FOR corrispondente
NOT INPUT FILE	Si è cercato di leggere dati da un file a sola scrittura

NOT OUTPUT FILE	Si è cercato di inviare dati ad un file a sola lettura
OUT OF DATA	Si è cercato di eseguire una READ senza che la corrispondente istruzione DATA contenga dati ancora non letti
OUT OF MEMORY	Il programma e le variabili hanno riempito l'intera memoria BASIC, oppure vi sono troppi cicli FOR-...NEXT nidificati, oppure infine sono stati eseguiti troppi GOSUB senza incontrare i relativi RETURN
OVERFLOW	Il risultato di un calcolo numerico è maggiore in valore assoluto del più alto valore memorizzabile, che è 1.7E+38
REDIM'D ARRAY	Si è cercato di eseguire una DIM su un vettore già dimensionato precedentemente
REDO FROM START	Durante l'esecuzione di una INPUT si è introdotta una stringa mentre era atteso un dato numerico; la INPUT viene rieseguita
RETURN WITHOUT GOSUB	Durante l'esecuzione del programma viene incontrato un RETURN non preceduto da un GOSUB
STRING TOO LONG	Più di 255 caratteri in una stringa
SYNTAX	Un'istruzione non è riconoscibile dall'Interprete BASIC (parole scritte in modo errato, parentesi mancanti, ecc.)
TYPE MISMATCH	C'è un dato numerico al posto di una stringa o viceversa
UNDEF'D FUNCTION	Si è cercato di utilizzare una funzione del tipo FNv senza averla precedentemente definita con DEF FNv=...
UNDEF'D STATEMENT	Il flusso del programma è stato diretto (tramite GOTO, GOSUB, ecc.) a una linea di programma non esistente
VERIFY	Il programma su nastro o disco non coincide con quello in memoria

SET DI CARATTERI

I codici di schermo 128–255 sono i reverse dei codici 0–127
I codici CBM ASCII 192–223 sono identici ai 96–127, i codici 224–254 sono identici ai 160–190, e il codice 255 è identico al 126
Gli altri codici CBM ASCII vanno ricercati tra i codici dei colori e di controllo

Codice schermo	Codice ASCII	Set n. 1	Set n. 2	Codice schermo	Codice ASCII	Set n. 1	Set n. 2
0	64	@	a	26	90	Z	
1	65	A	b	27	91	[	
2	66	B	c	28	92	£	
3	67	C	d	29	93	]	
4	68	D	e	30	94	↑	
5	69	E	f	31	95	←	
6	70	F	g	32	32		
7	71	G	h	33	33	!	
8	72	H	i	34	34	"	
9	73	I	j	35	35	#	
10	74	J	k	36	36	$	
11	75	K	l	37	37	%	
12	76	L	m	38	38	&	
13	77	M	n	39	39	'	
14	78	N	o	40	40	(	
15	79	O	p	41	41	)	
16	80	P	q	42	42	*	
17	81	Q	r	43	43	+	
18	82	R	s	44	44	,	
19	83	S	t	45	45	-	
20	84	T	u	46	46	.	
21	85	U	v	47	47	/	
22	86	V	w	48	48	0	
23	87	W	x	49	49	1	
24	88	X	y	50	50	2	
25	89	Y	z	51	51	3	

Codice schermo	Codice ASCII	Set n. 1	Set n. 2
52	52	4	
53	53	5	
54	54	6	
55	55	7	
56	56	8	
57	57	9	
58	58	:	
59	59	;	
60	60	<	
61	61	=	
62	62	>	
63	63	?	
64	96	─	
65	97	♠	A
66	98	│	B
67	99	─	C
68	100	─	D
69	101	─	E
70	102	─	F
71	103	│	G
72	104	│	H
73	105	╮	I
74	106	╰	J
75	107	╯	K
76	108	└	L
77	109	╲	M
78	110	╱	N
79	111	┌	O
80	112	┐	P
81	113	●	Q
82	114	─	R
83	115	♥	S
84	116	│	T
85	117	╭	U
86	118	╳	V
87	119	○	W
88	120	♣	X
89	121	│	Y
90	122	◆	Z
91	123	┼	

Codice schermo	Codice ASCII	Set n. 1	Set n. 2
98	124	▒	
99	125	▕	
100	126	π	▒
101	127	◥	▧
102	160		
103	161	▌	
104	162	▄	
105	163	▔	
106	164	▁	
107	165	▏	
108	166	▒	
109	167	▕	
110	168	▒	
111	169	◤	▨
112	170	▕	
113	171	├	
114	172	▗	
115	173	└	
116	174	┐	
117	175	▂	
118	176	┌	
119	177	┴	
120	178	┬	
121	179	┤	
122	180	▎	
123	181	▍	
124	182	▊	
125	183	▔	
126	184	▃	
127	185	▂	
	186	┘	✓
	187	▖	
	188	▝	
	189	┘	
	190	▘	
	191	▚	

CODICI DEI COLORI

Colore	Codice	Carattere	ASCII	Tasto
Nero	0		144	CTRL 1
Bianco	1	E	5	CTRL 2
Rosso	2		28	CTRL 3
Ciano	3		159	CTRL 4
Porpora	4		156	CTRL 5
Verde	5		30	CTRL 6
Blu	6		31	CTRL 7
Giallo	7		158	CTRL 8
Arancio	8		129	C 1
Marrone	9		149	C 2
Rosa	10		150	C 3
Grigio scuro	11		151	C 4
Grigio	12		152	C 5
Verde chiaro	13		153	C 6
Blu chiaro	14		154	C 7
Grigio chiaro	15		155	C 8

CODICI DI CONTROLLO

Funzione	Carattere	ASCII	Tasto
Attiva reverse	R	18	CTRL 9

Disattiva reverse		146	CTRL 0
Cursore giù		17	↑CRSR↓
Cursore su		145	SHIFT ↑CRSR↓
Cursore destra		29	←CRSR→
Cursore sinistra		157	SHIFT ←CRSR→
Cursore in alto a sinistra		19	CLR/HOME
Canc. schermo-Curs. in alto a sin.		147	SHIFT CLR/HOME
Tasto funzione 1		133	f1
Tasto funzione 2		137	SHIFT f1
Tasto funzione 3		134	f3
Tasto funzione 4		138	SHIFT f3
Tasto funzione 5		135	f5
Tasto funzione 6		139	SHIFT f5
Tasto funzione 7		136	f7
Tasto funzione 8		140	SHIFT f7

ALTA RISOLUZIONE

* Per visualizzare la pagina grafica: POKE 53265, PEEK (53265) OR 32
* Per tornare allo schermo normale: POKE 53265, PEEK (53265) AND 223

La pagina grafica occupa 8000 byte, cioè 64000 bit, ognuno dei quali corrisponde ad un pixel, che risulta acceso o spento a seconda che il bit di cui è l'immagine valga 1 oppure 0. Normalmente la pagina grafica occupa l'area RAM che va dalla locazione 8192 alla 16191. L'informazione riguardo al colore dei pixel e dello sfondo non è contenuta nella mappa-colori (55296–56295) ma bensì nella mappa-caratteri (1024–2023). Per l'esattezza, i 4 bit meno significativi di ogni byte danno il colore dello sfondo, mentre i 4 bit più significativi danno il colore dei pixel accesi; ad esempio, il valore 2*16+6=38 corrisponde a pixel rossi (2) su sfondo blu (6). Per indirizzare un singolo pixel usiamo un sistema di coordinate (X,Y) avente come origine l'angolo superiore sinistro dello schermo (X varia tra 0 e 319, Y tra 0 e 199). Per operare sul pixel di coordinate X, Y dobbiamo calcolare i seguenti valori:

```
RIGA = INT(Y/8)
COLONNA = INT(X/8)
LINEA = Y AND 7
BIT = 7 – (X AND 7)

BYTE = 8192 + RIGA*320 + COLONNA*8 + LINEA
```

A questo punto per accendere il pixel è sufficiente l'istruzione

```
POKE BYTE, PEEK (BYTE) OR 2↑BIT
```

Per spegnerlo invece si deve usare

```
POKE BYTE, PEEK (BYTE) AND (255 – 2↑BIT)
```

REGISTRI DEL SUONO

Registro	Funzione	Locazione
	VOCE 1	
0	Byte basso frequenza	54272
1	Byte alto frequenza	54273
2	Byte basso largh. onda rett.	54274
3	4 bit alti largh. onda rett.	54275
4	Registro di controllo	54276
5	Attack/Decay	54277
6	Sustain/Release	54278
	VOCE 2	
7	Byte basso frequenza	54279
8	Byte alto frequenza	54280
9	Byte basso largh. onda rett.	54281
10	4 bit alti largh. onda rett.	54282
11	Registro di controllo	54283

12	Attack/Decay	54284
13	Sustain/Release	54285
	VOCE 3	
14	Byte basso frequenza	54286
15	Byte alto frequenza	54287
16	Byte basso largh. onda rett.	54288
17	4 bit alti largh. onda rett.	54289
18	Registro di controllo	54290
19	Attack/Decay	54291
20	Sustain/Release	54292
	FILTRI	
21	3 bit bassi frequenza taglio	54293
22	Byte alto frequenza taglio	54294
23	Risonanza/Sel. voce filtrata	54295
24	Sel. filtri/Volume	54296
	REGISTRI A SOLA LETTURA	
27	Valore oscillatore voce 3	54299
28	Valore inviluppo voce 3	54300

USO DEL SUONO

Per produrre un suono con la voce 1:

—Regolare il volume V (0–15): POKE 54296, V
—Regolare l'attack A (0–15) e il decay D (0–15): POKE 54277, A*16 + D
—Regolare il sustain S (0–15) e il release R (0–15): POKE 54278, S*16 + R
—Regolare la frequenza F (0–65535), che divisa in byte alto (FA) e byte basso (FB), secondo la formula FA=INT(F/256) e FB=F–FA:
POKE 54272, FB : POKE 54273, FA
—Se si usa la forma d'onda rettangolare, regolare la larghezza L dell'impulso (0–4096), che è divisa in 4 bit alti (LA) e un byte basso (LB), secondo la formula LA=INT(L/256) e LB=L–LA:
POKE 54274, LB : POKE 54275, LA
L'onda quadra si ottiene per L=2048 (LA=8 e LB=0)
—Attivare la voce selezionando la forma d'onda W (4–7) : POKE 54276, 2↑W + 1
Per W=4 abbiamo onda triangolare, per W=5 onda dente-di-sega, per W=6 onda rettangolare e per W=7 rumore.

VALORI DELLE NOTE MUSICALI

Ecco una tabella contenente i valori (byte alto/byte basso) di frequenza per ottenere 7 ottave complete di note musicali:

Nota	Ott. 1		Ott. 2		Ott. 3		Ott. 4		Ott. 5		Ott. 6		Ott. 7	
DO	1	18	2	37	4	73	8	143	17	37	34	75	68	149
DO#	1	35	2	69	4	139	9	21	18	42	36	85	72	169
RE	1	52	2	104	4	208	9	159	19	63	38	126	76	252
RE#	1	70	2	140	5	25	10	60	20	100	40	200	81	161
MI	1	90	2	179	5	103	10	205	21	154	43	52	86	105
FA	1	110	2	220	5	185	11	114	22	227	45	198	91	140
FA#	1	132	3	8	6	16	12	32	24	63	48	127	96	254
SOL	1	155	3	54	6	108	12	216	25	177	51	97	102	194
SOL#	1	179	3	103	6	206	13	156	27	56	54	111	108	223
LA	1	205	3	155	7	53	14	107	28	214	57	172	115	88
LA#	1	233	3	210	7	163	15	70	30	141	61	126	122	52
SI	2	6	4	12	8	23	16	47	32	94	64	188	129	120

REGISTRI DELLE SPRITE

Registro	Funzione	Locazione
0	Coordinata X sprite 0	53248

1	Coordinata Y sprite 0	53249
2	Coordinata X sprite 1	53250
3	Coordinata Y sprite 1	53251
4	Coordinata X sprite 2	53252
5	Coordinata Y sprite 2	53253
6	Coordinata X sprite 3	53254
7	Coordinata Y sprite 3	53255
8	Coordinata X sprite 4	53256
9	Coordinata Y sprite 4	53257
10	Coordinata X sprite 5	53258
11	Coordinata Y sprite 5	53259
12	Coordinata X sprite 6	53260
13	Coordinata Y sprite 6	53261
14	Coordinata X sprite 7	53262
15	Coordinata Y sprite 7	53263
16	Bit alti coordinate X	53264
21	Accende/spegne sprite	53269
23	Espansione verticale	53271
27	Priorità sprite-caratteri	53275
28	Seleziona sprite multicolor	53276
29	Espansione orizzontale	53277
30	Collisione sprite-sprite	53278
31	Collisione sprite-caratteri	53279
37	Registro multicolor 1	53285
38	Registro multicolor 2	53286
39	Colore sprite 0	53287
40	Colore sprite 1	53288
41	Colore sprite 2	53289
42	Colore sprite 3	53290
43	Colore sprite 4	53291
44	Colore sprite 5	53292
45	Colore sprite 6	53293
46	Colore sprite 7	53294

USO DEGLI SPRITE

Ogni sprite è largo 24 pixel e alto 21, e occupa quindi 63 byte che possiamo pensare disposti in 21 righe da 3 byte. Come nella pagina grafica, per accendere un pixel dobbiamo porre a 1 il bit corrispondente. Agli 8 sprite corrispondono altrettanti puntatori, che si trovano nelle locazioni 2040–2047 e servono a indicare al C64 le aree di memoria occupate dagli sprite. Ad esempio:

```
POKE 2040+S, 192+S
```

dove S è il numero dello sprite (0–7), comunica al computer che lo sprite 0 si trova all'indirizzo 192*64 = 12288, occupando quindi l'area 12288–12350; lo sprite 1, invece, si trova all'indirizzo 193*64 = 12352 e occupa l'area 12352–12414, e cosi via.
Il colore di ogni sprite viene determinato con

```
POKE 53287+S, C
```

dove C è il codice del colore.
Nei registri 16–21–23–27–28–29 ognuno degli 8 bit corrisponde a uno sprite nel modo seguente:

Numero sprite	0	1	2	3	4	5	6	7
Valore del bit	1	2	4	8	16	32	64	128

Se ad esempio vogliamo espandere verticalmente uno sprite, dobbiamo usare

```
POKE 53271, v
```

dove V = 2↑S è il valore del bit corrispondente.
Esiste la possibilità di testare, tramite i registri 30 e 31, eventuali collisioni tra due sprite oppure tra uno sprite e un carattere. Nel primo caso il registro 30 contiene la somma dei valori dei 2 bit corrispondenti ai 2 sprite che sono venuti a contatto, mentre nel secondo caso il registro 31 contiene il valore del bit corrispondente allo

sprite che ha toccato il carattere. Questi, registri si azzerano dopo essere letti tramite una istruzione PEEK, in modo da essere pronti per un nuovo test.
La posizione di ogni sprite è determinata dalle coordinate X,Y.
Y varia tra 0 e 255, ma solo i valori 50-249 si trovano sullo schermo visibile.
Per posizionare verticalmente uno sprite si usa quindi

```
POKE 53249 + 2*S, Y
```

X invece varia tra 0 e 511, ma solo i valori 24-343 si trovano sullo schermo visibile. Per esprimere numeri superiori a 255 occorrono 9 bit, cioè un byte più un bit che si trova nel registro 16. Per posizionare orizzontalmente uno sprite, con X<=255, si usa

```
POKE 53248 + 2*S, X
```

Se invece X>255 la procedura è la seguente

```
POKE 53264, V
POKE 53248 + 2*S, X−256
```

SUDDIVISIONE DELLA MEMORIA

Ind.dec	Contenuto	Tipo	Ind.esadec.
0	RAM USATA DA INT.BASIC E KERNAL	(RAM)	0000
1023	BUFFER REGISTRATORE		03FF
1024	MAPPA SCHERMO	(RAM)	0400
2047	PUNTATORI SPRITE		07FF
2048	AREA BASIC	(RAM)	0800
	Programma Utente		
	Variabili		
	Vettori		
	Memoria libera		
40959	Stringhe		9FFF
40960	INTERPRETE BASIC	(ROM)	A000
49151	(sotto: RAM)		BFFF
49152	RAM UTENTE	(RAM)	C000
53247			CFFF
53248	AREA INPUT/OUTPUT PER VIC E SID	(Registri)	D000
55295	(sotto: ROM caratteri o RAM)		D7FF
55296	MAPPA COLORE	(RAM)	D800
56319	(sotto: ROM caratteri)		DBFF
56320	AREA INPUT/OUTPUT PER CIA	(Registri)	DC00
57343	(sotto: ROM caratteri o RAM)		DFFF
57344	SISTEMA OPERATIVO KERNAL	(ROM)	E000
65535	(sotto: RAM)		FFFF

LOCAZIONI FONDAMENTALI

Locazione	Funzione
197	Lettura tastiera
53280	Colore bordo
53281	Colore sfondo
54297	Lettura potenziometro paddle 1
54298	Lettura potenziometro paddle 2
56320	Lettura joystick 1 e pulsante paddle
56321	Lettura joystick 2 e pulsante paddle

NOTA: Queste locazioni sono a sola lettura, tranne i due registri-colore che sono invece a sola scrittura